어드벤처디자인 기초 레벨

세그먼트 트리를 사용한

영역 합의 질의 문제

|  |  |
| --- | --- |
| 학과 | 컴퓨터공학과 |
| 학번 | 20221049 |
| 학년 | 2 |
| 이름 | 정보경 |
| 제출 일자 | 2023. 09. 22. |

목차

1. 문제 개요
2. 문제 해결 방법
3. 알고리즘 분석
4. 구현 및 실험 결과
5. 기타 사항

[별첨] 소스코드

1. 문제 개요

n개의 정수 값을 갖는 배열 list[0], list[1], list[2], … , list[n-1]이 주어질 때 주어진 list 배열에서 인덱스 s에서 e까지의 배열 원소들의 합을 구한다. 여기서 인덱스 s에서 e의 범위는 0≤s≤e≤n-1이 된다. 이러한 문제를 영역 질의 합의(range sum query) 문제라 한다.

그러나 이 문제를 단순하게 s부터 e까지 반복문을 통해 더하게 되면 수행시간은 O(n)이 된다. 따라서 이 문제의 수행시간을 개선하기 위한 자료구조를 채택하여야 한다

2. 문제 해결 방법

수행시간을 개선하기 위한 자료구조로 세그먼트 트리를 사용한다. 세그먼트 트리는 배열로 주어진 입력 리스트의 원소들의 부분 합을 이진 트리 형식으로 구성한 자료구조이다. 입력된 리스트를 반으로 쪼개어 두 개의 리스트로 분할하고, 또 그 각각의 분할된 리스트를 다시 두 개의 리스트로 분할한다. 마지막에 각 원소로 모두 쪼개져 더 이상 분할할 수 없는 상태가 되면, 각 원소들은 단말 노드가 되어 트리를 구성한다. 이렇게 구성된 단말 노드부터 시작하여 부모 노드는 자식 원소의 합이 되며 트리의 비 단말 노드를 구성해 나간다. 즉, 부모 노드는 자식 노드의 합이고, 그 부모 노드 또한 해당 노드의 형제 노드와의 합이 부모 노드로 저장되는 형태를 이루는 것이다. 또한 세그먼트 트리는 배열을 사용한다. 일반적으로, 완전이진트리에서 배열을 사용하고, 일반적인 이진트리에서는 연결리스트를 사용한다. 세그먼트 트리는 완전이진트리는 아니지만, 매번 반으로 쪼개어 트리를 구성하므로 균형 잡힌 트리라 볼 수 있다. 따라서 배열을 사용하여 구성한다.

이러한 세그먼트 트리를 사용하기 위해 아래 세 가지의 알고리즘을 설계한다.

* 입력 리스트의 배열을 세그먼트 트리로 구성한다.
* 세그먼트 트리를 사용하여 질의 범위의 합을 계산한다.
* 입력 리스트의 배열이 변경될 경우 세그먼트 트리를 갱신한다.

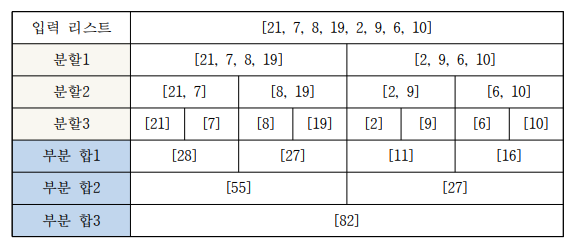
3. 알고리즘 분석

(1) 세그먼트 트리의 구성

주어진 입력 리스트를 같은 크기 혹은 한 개의 원소만큼만 차이나는 크기의 두 리스트로, 나눈 리스트를 마찬가지로 두 개의 리스트로 분할한다. 이러한 분할은 최종적으로 더 이상 분할 되지 않는 상태, 즉 한 개의 원소만 남아있을 때까지 반복한다.

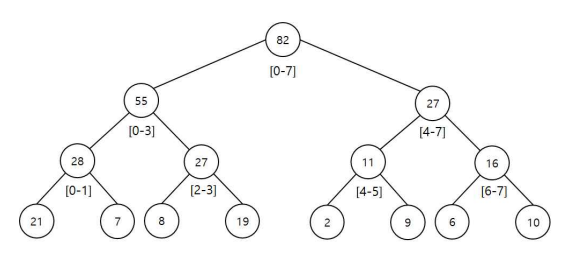
분할이 끝나면 부분합을 구성한다. 분할되어 한 개의 원소로 이루어진 리스트들을 단말 노드로 구성한 상태로, 비 단말 노드는 자신의 자식의 합이 되도록 구성한다. 형제 노드와의 합을 부모 노드에 기록하고, 그 부모 노드 또한 형제 노드와의 합을 자신의 부모 노드에 기록하며 루트 노드까지 반복하게 된다. 즉 단말 노드에서부터 루트 노드까지 반복하여 합을 채워 넣는 방식으로 구성하게 된다. (표 1.1)은 이러한 세그먼트 트리의 구성 과정을 보여준다.

(표 1.1) 입력 리스트를 분할한 후에 세그먼트 트리를 구성하는 과정



(그림 1.2)는 (표 1.1)의 구성 과정을 통해 만들어진 세그먼트 트리이다. 단말 노드는 입력 리스트의 각 원소들이 배치되었으며, 비 단말 노드는 각각의 두 자식의 합이 저장되었다. 세그먼트 트리를 구성하기 위해 리스트의 분할과 부분 합의 계산을 위한 두 개의 단계로 구분되며, 입력 리스트에 포함된 데이터의 수가 n일 경우 각각 O(logn)번이 반복 수행된다.

(그림 1.2) 8개의 입력 리스트에 대한 세그먼트 트리의 구성



(알고리즘 Seg\_tree\_const)는 n개의 정수형 데이터를 갖는 입력 리스트가 배열 list로 주어질 경우 (알고리즘 Construct\_tree)를 호출하여 세그먼트 트리를 구성하는 알고리즘이다. 알고리즘의 입력은 n개의 정수 갑이 저장된 list 배열이고 출력은 세그먼트 트리가 구성된 배열 seg\_tree이다.

1. Algorithm Seg\_tree\_const(list, n)
2. Input: n개의 정수형 데이터를 갖는 입력 리스트 배열(list)
3. Output: 입력 리스트의 세그먼트 트리를 저장한 배열(seg\_tree)
4. height <- ⌈logn⌉
5. t\_size <- 2 × 2height - 1
6. seg\_tree <- malloc(t\_size)
7. Construct\_tree(list, 0, n-1, seg\_tree, 0)
8. return seg\_tree

(알고리즘 Construct\_tree)는 입력 리스트를 반복적으로 분할하고 마지막으로 분할된 리스트로부터 부분 합을 계산하여 세그먼트 트리를 구성하는 알고리즘이다. 알고리즘의 입력에서 start와 end는 가장 최근에 분할된 리스트의 시작과 끝 인덱스 값을 의미한다. Current는 재귀적으로 반복되는 과정에서 부분 합을 계산하기 위해 현재의 상태에서 세그먼트 트리의 배열 seg\_tree의 인덱스 값을 의미한다.

1. Algorithm Construct\_tree(list, start, end, seg\_tree, current)
2. Input: 입력 리스트 배열(list), 분할된 리스트의 시작(start)과 끝(end), 현재 노드의 인덱스(current), 세그먼트 트리 배열(seg\_tree)
3. Output: 세그먼트 트리의 현재 인덱스 배열
4. if (start = end) {
5. seg\_tree[current] <- list[start]
6. return list[start] }
7. mid <- start + (end – start) / 2
8. child <- 2 \* current
9. seg\_tree[current] <- Construct\_tree(list, start, mid, seg\_tree, child+1) + Construct\_tree(list, mid+1, end, seg\_tree, child+2)
10. return seg\_tree[current]

(2) 세그먼트 트리를 사용하여 질의 영역의 합 계산

주어진 입력 리스트에 대해 (1)에서 설명한 알고리즘에 따라 구성된 세그먼트 트리를 사용하여 질의 영역의 합을 계산하는 알고리즘을 설계한다.

(알고리즘 Get\_query)는 입력 리스트에서 질의 범위로 주어진 합을 계산하여 반환하는 알고리즘이다. 알고리즘에서 입력으로 (1)에서 부분 합이 계산된 세그먼트 트리 배열(seg\_tree)과 부분 합을 계산하기 위한 질의의 시작(q\_s)과 끝(q\_e)인덱스가 주어진다. 질의로 주어진 인덱스의 범위가 올바른지 확인하여 오류 처리를 하고 종료하는 것이 4번 줄에 나타나있다. 5번 줄에서는 (알고리즘 Query\_sum)을 호출하여 부분합의 계산 결과를 저장하고, 이를 반환에 사용한다.

1. Algorithm Get\_query(seg\_tree, n, q\_s, q\_e)
2. Input: 세그먼트 트리 배열(seg\_tree), 부분 합을 계산하기 위한 시작과 끝 인덱스 위치(q\_s, q\_e)
3. Output: 질의한 위치의 부분 합
4. if (q\_s < 0 or q\_e > n-1 or q\_e > q\_s) return error
5. sum <- Query\_sum(seg\_tree, 0, n-1, q\_s, q\_e, current)
6. return sum

(알고리즘 Query\_sum)은 입력 리스트로 구성된 세그먼트 트리의 배열을 사용하여 질의 범위의 부분 합을 계산하는 알고리즘이다. 이 알고리즘은 질의 범위의 부분 합을 계산하기 위해 다음과 같이 세 가지 경우로 구분되어 수행된다.

* 질의한 인덱스의 시작과 끝의 범위가 현재 분할된 리스트의 시작과 끝의 인덱스 범위를 포함하는 경우 세그먼트 트리 배열에서 현재 위치의 부분 합인 seg\_tree[current] 값을 반환한다(4번 줄).
* 질의한 인덱스의 시작과 끝의 범위가 현재 분할된 리스트의 시작과 끝의 인덱스 범위를 벗어나는 경우 0을 반환한다(5번 줄).
* 위의 두 경우에 해당하지 않을 경우 분할된 두 리스트의 부분 합을 순환적으로 호출하여 계산한다(6~8번 줄).

1. Algorithm Query\_sum(seg\_tree, start, end, q\_s, q\_e, current)
2. Input: 세그먼트 트리 배열(seg\_tree), 분할된 리스트의 시작과 끝 인덱스(start/end), 질의 인덱스(q\_s/q\_e), 현재 노드 인덱스(current)
3. Output: 질의 범위의 부분 합
4. if (q\_s <= start and q\_e >= end) return seg\_tree[current]
5. if (end < q\_s or start > q\_e) return 0
6. mid <- start + (end – start) / 2
7. child <- 2 \* current
8. return Query\_sum(seg\_tree, start, mid, q\_s, q\_e, child+1) + Query\_sum(seg\_tree, mid+1, end, q\_s, q\_e, child+2)

(3) 세그먼트 트리의 갱신

(삼성노트와 책)

4. 구현 및 실험 결과

5. 기타 사항

   - 문제 개요

   - 문제 해결 방법 (자료구조, 알고리즘 등)

   - 알고리즘 분석 (시간, 공간 등)

   - 구현 및 실험 결과

   - 기타 사항

   - 별첨 : 소스프로그램

맞춤법 검사기 한번 돌려서 오타 확인!